PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 | A1

(11) International Publication Number: WO 97/46956

(43) International Publication Date: 11 December 1997 (11.12.97)

(21) International Application Number: PCT/US97/09295

(22) International Filing Date: 6 June 1997 (06.06.97)

(30) Priority Data:
60/019,303 7 June 1996 (07.06.96) US

(71) Applicant: AT & T CORP. [US/US]; 32 Avenue of the Americas, New York, NY 10013-2412 (US).

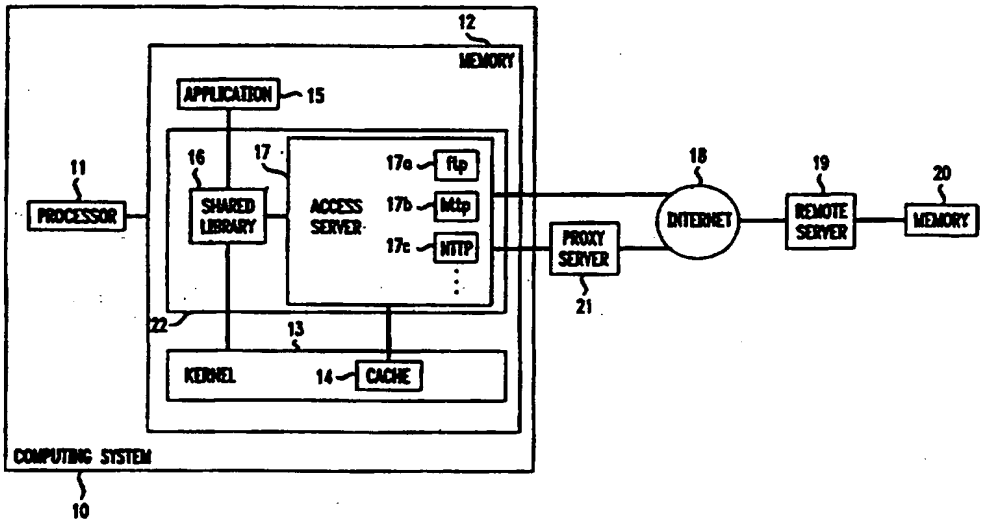
(72) Inventor: RAO, Chung-Hwa, Herman; 4304 Springbrook Drive, Edison, NJ 08820 (US).

(74) Agents: DWORETSKY, Samuel, H. et al.; AT & T Corp., P.O. Box 4110, Middletown, NJ 07748 (US).

(81) Designated States: CA, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: INTERNET FILE SYSTEM



(57) Abstract

A method and a system in which a computing system transparently accesses resources connected to the Internet. A memory of the computing system contains an operating system, a cache associated with the operating system, a shared library and an access server. The shared library is responsive to a system call for a file by determining whether a path name for the file is located under a personal name space in the memory and by issuing a request for retrieving the file from an Internet resource based on the path name located under the personal name space when the file is not stored in the cache and has a path name located under the personal name space. An access server, in response to the request from the shared library, selects an appropriate access protocol for retrieving the file from the Internet resource and retrieves the file from the Internet resource. The shared library then issues the system call to the operating system when the access server retrieves the file. The access server restores the file to the Internet resource when the application closes the file.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

FIG. 1

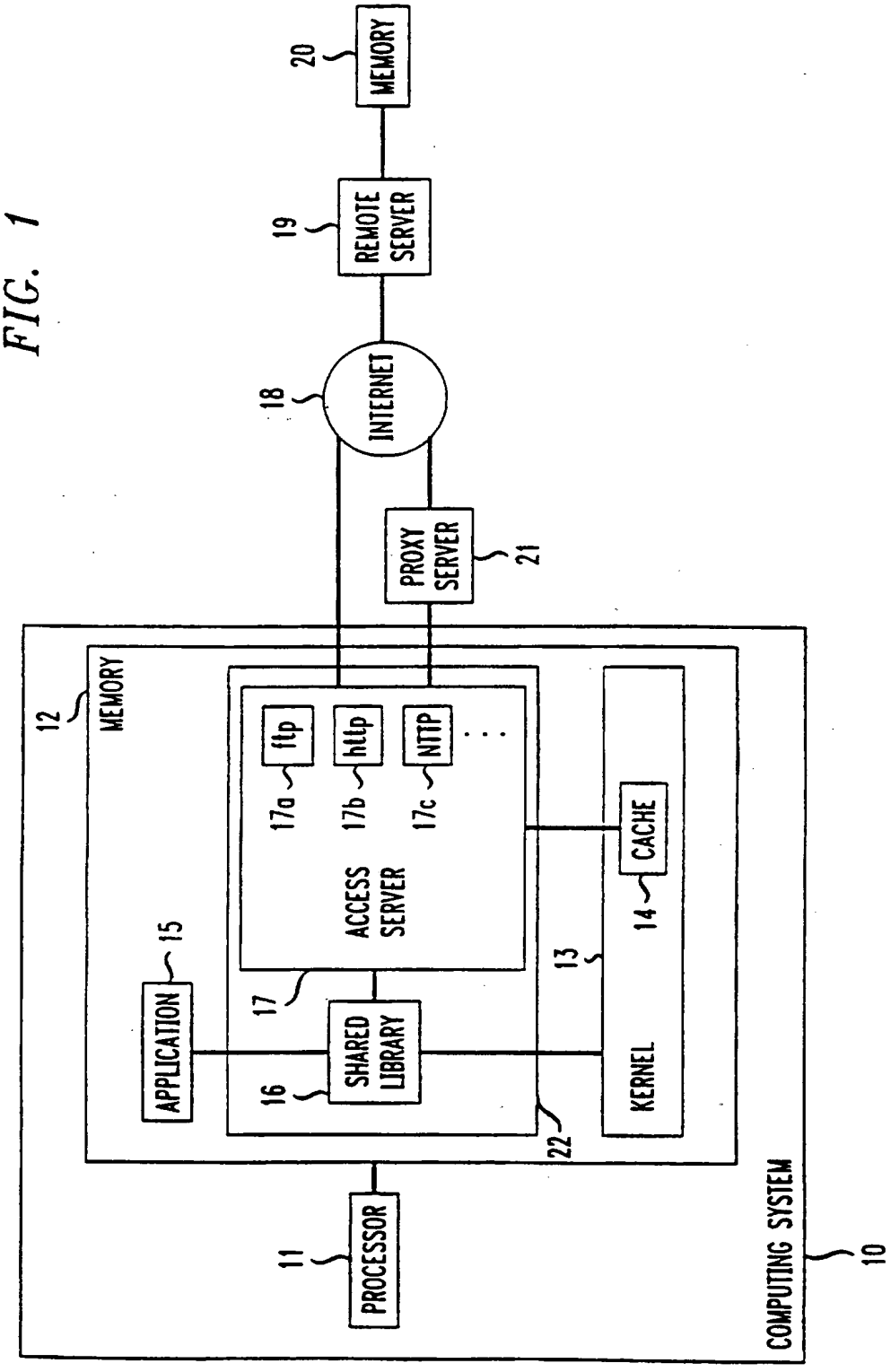


FIG. 2

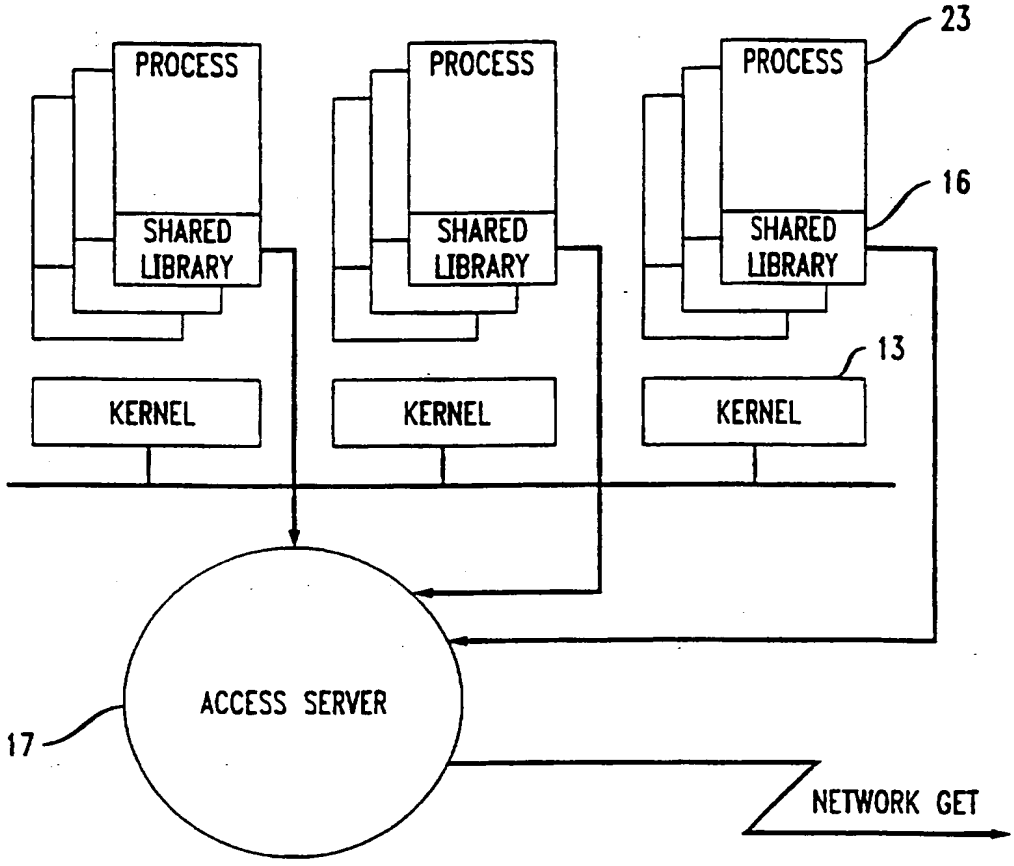


FIG. 3

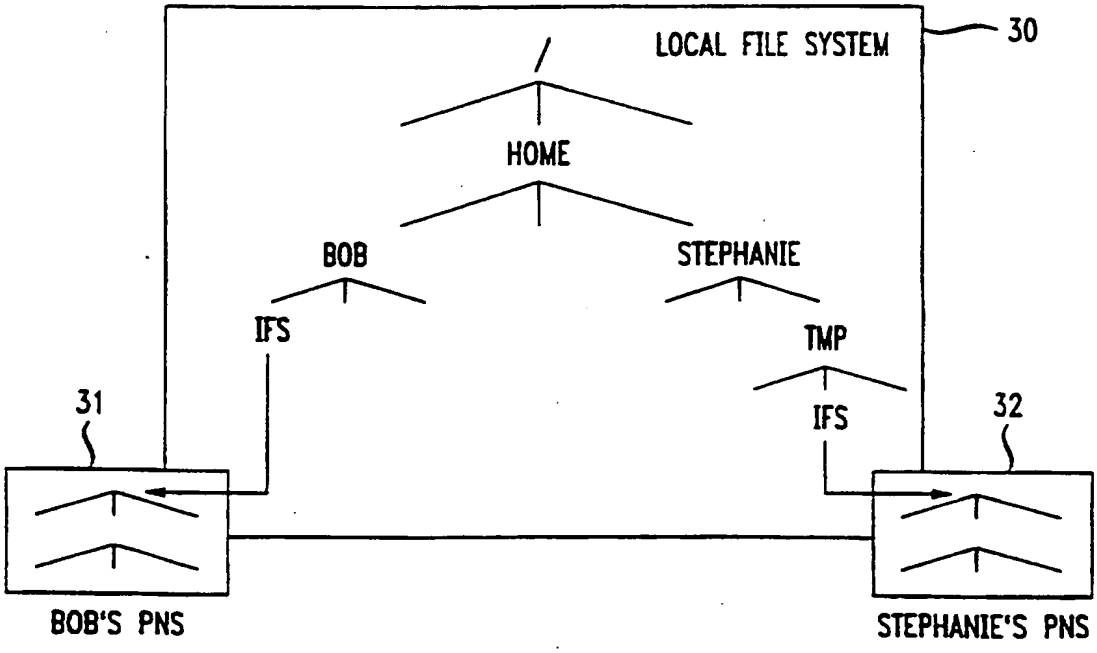


FIG. 4

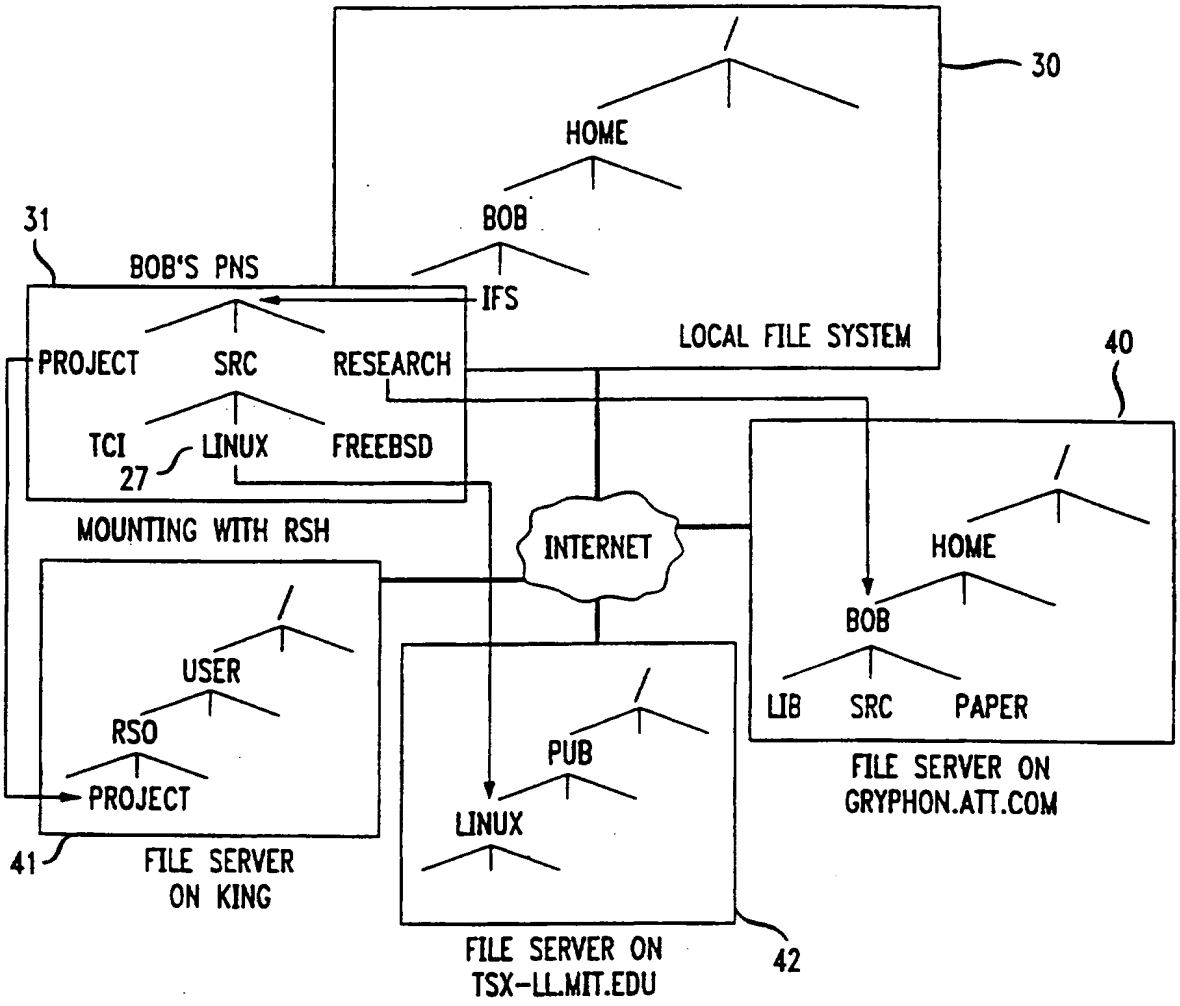


FIG. 5

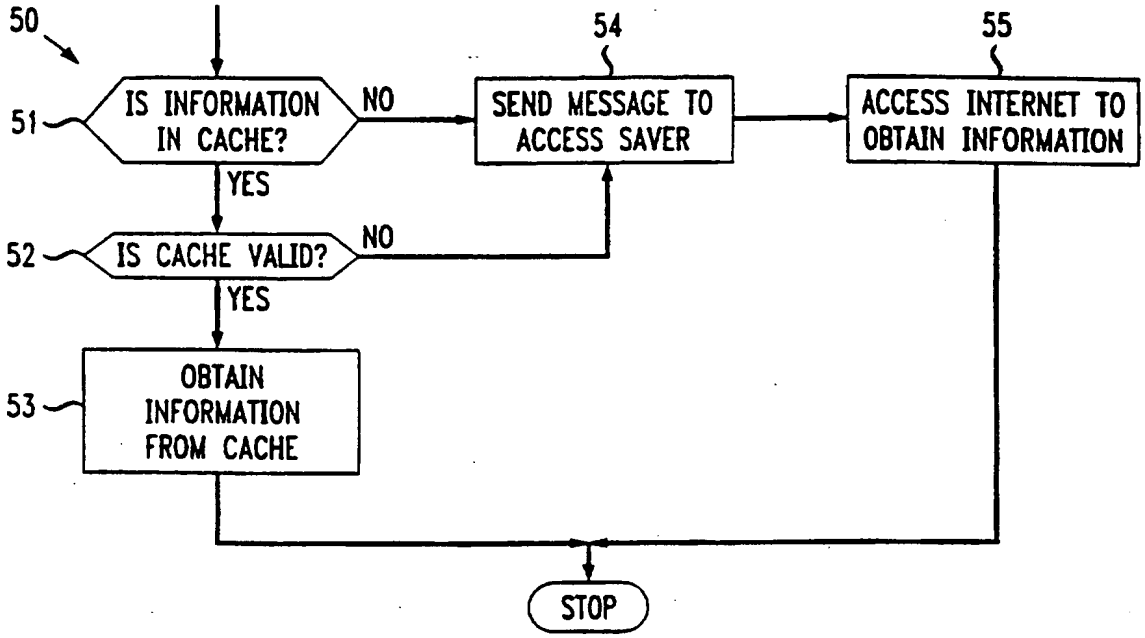
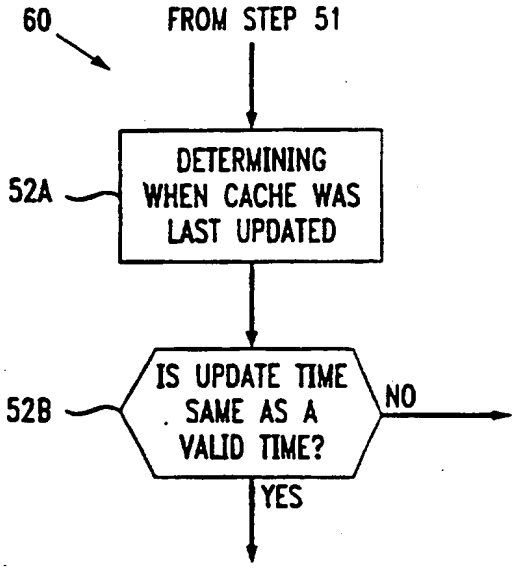


FIG. 6



INTERNET FILE SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims priority to United States provisional patent application Serial No. 60/019,303 to C.H. Rao, filed June 7, 1996, and entitled "Integrating The Internet Into A File System."

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of computing. More particularly, the present invention relates to a method and a system for accessing resources that are available over the Internet.

2. Description of the Related Art

As more resources become available over the Internet, it is becoming increasingly difficult to locate, manage and integrate resources such as Web pages, Gopher information, Network News, released documents and technical reports, public domain software, and collaborative sources and drafts. Resources connected to the Internet are accessed using a naming scheme that is defined by the Uniform Resource Locator (URL) protocol, which has become a de facto standard. Internet resources are also accessed using other standard access protocols, such as the File Transfer Protocol (FTP) for file servers, the Hypertext Transport Protocol (HTTP) for web servers, the gopher protocol

(GOPHER) for gopher servers, the Network News Transfer Protocol (NNTP) for news servers, and the Remote Shell (RSH) for file servers. Each of these particular access protocols define access mechanisms for retrieving resources from servers that are connected to the Internet. Internet browsers, such as Mosaic and Netscape, have been introduced for conveniently searching networks and retrieving information, but while perfectly adequate for searching and retrieving resources, such conventional browsers are limited and do not lend themselves for integration into larger toolsets.

Several systems have been developed for making access to files over the Internet easier and more efficient. For example, the Andrew File System provides a wide area file service by extending the scope of file systems from a Local Area Network (LAN) to the Internet. However, the Andrew File System uses its own protocol, not the standard Internet protocols previously mentioned and, consequently, requires an Andrew file server.

Another system that has been developed for accessing files over the Internet is the Bell Laboratory Plan 9 which permits mounting of Internet file servers to local name spaces by introducing a new operating system. Plan 9 supports a per-process name space and a message-oriented file system protocol so that a file system-like interface is implemented. While different access protocols are accommodated, the Plan 9 operating system is not transparent to existing application tools and kernels and the applications running on the kernels require modification for use with Plan 9.

The Alex File System provides transparent read access to Internet resources by being built on top of the existing Network File System (NFS), with an Alex server being added as a Virtual File System (VFS) interface. Hosts on a LAN use the NFS protocol for sending calls to the Alex server. The calls pass through an NFS client on a kernel that is local to the NFS server kernel on which the Alex server is running. The calls are then upcalled from the NFS server kernel to the Alex server. Nevertheless, the NFS protocol causes the Alex system to be limited because, while the NFS protocol implements file services, no information has been provided regarding a process initiating a request, which is important for implementing an authenticated access. Consequently, the Alex system only supports limited file-oriented file access, such as an anonymous FTP access.

The Jade File System provides a uniform way for naming and accessing files in an Internet environment using a new personal name server that integrates a heterogeneous collection of underlying existing file systems that may not be modified because of autonomy. A private name space can be defined by each user that supports two features: multiple file systems are allowed to be mounted under one directory, and one logical name space is permitted for mounting other logical name spaces. However, the Jade File System does not use URL naming. Instead, a separate name server is provided for a pathname. Further, individual access servers must be mounted and the program running on the host must be recompiled before a user can access a file system.

The Multiple Dimensional File System (n-DFS) is a logical file system that allows new services to be added to underlying file systems without requiring applications or the operating system kernel to be modified. The n-DFS is layered on top of existing physical file systems, but does not provide convenient Internet access using existing protocols.

What is needed is an Internet file system that transparently uses existing protocols in connection with existing operating system kernels, applications and file servers for accessing Internet resources.

SUMMARY OF THE INVENTION

The present invention provides an Internet file system that transparently uses existing protocols in connection with existing operating system kernels, applications and file servers for accessing Internet resources. Consequently, an application programmer can use the same system calls to handle files, whether stored locally or at Internet resources. The advantages of the present invention are provided by a method and a system in which a computing system having a processor and a memory transparently accesses resources connected to the Internet. The memory of the computing system contains an operating system, a cache associated with the operating system, a shared library and an access server. The shared library is responsive to a system call for a file by determining whether a path name for the file is located under a personal name space in the memory and by issuing a request for retrieving the file from an Internet resource based on the path name located

under the personal name space when the file is not stored in the cache and has a path name located under the personal name space In response to the request from the shared library, the access server selects an appropriate access protocol for retrieving the file from the Internet resource and retrieves the file from the Internet resource. The shared library then issues the system call to the operating system when the access server retrieves the file. The access server restores the file to the Internet resource when the application closes the file.

BRIEF DESCRIPTION OF THE DRAWING

The present invention is illustrated by way of example and not limitation in the accompanying figures in which like reference numerals indicate similar elements and in which:

Figure 1 is a block diagram of the architecture of a system according to the present invention;

Figure 2 is a block diagram of a system having an architecture that supports coherent sharing for different applications according to the present invention;

Figure 3 illustrates the concept of a personal name space (PNS) according to the present invention;

Figure 4 shows a diagram illustrating the mounting of remote file systems according to the present invention;

Figure 5 shows a flow diagram of a validation process according to the present invention; and

Figure 6 shows a flow diagram of the details of a step of the validation process of Figure 5.

DETAILED DESCRIPTION

The present invention relates to an Internet File System (IFS) that allows users to manage Internet files in a Personal Name Space (PNS) by transparently integrating existing protocols, while requiring no modification of software or change in management of remote servers. The present invention introduces a logical layer between an operating system and applications running on the operating system that presents the applications with exactly the same system call interface that the underlying operating system provides. The logical layer of the present invention runs in the same address space as the application without requiring operating system modifications.

According to the invention, each user selects Internet files for access by organizing the files within a PNS and thereby avoids the complexity of maintaining a system-wide global name space. According to the present invention, a PNS is private and remains undefined to all other users. Only the owner the PNS has access to the PNS. A PNS can be "mounted" or attached to a shared name space or to another name space that supports URL naming. Internet files can be accessed directly using the URL names of the files as pathnames. For example, a home page identified by http://www.att.com/research.html is named ~/IFS/http:/www.att.com/research.html, where ~/IFS is the mount point on a

local file system for a PNS. Additionally, a PNS supports mount operations that attach name spaces of Internet file servers to nodes of the PNS using access protocols, such as File Transport Protocol (FTP). Thus, a PNS allows a file system to name and access files, and to perform mounting operations that support location-independent naming.

Figure 1 is a block diagram of the architecture of a system according to the present invention. System 10 includes a processor 11 coupled to a memory 12. Memory 12 includes memory space for storing an operating system kernel 13, a cache 14 for kernel 13, at least one application 15, such as a browser, a shared library 16 and an access server 17. Processor 11 and memory 12 can be physically located within a single device or can be configured as a plurality of processors and memory devices that are distributed using well-known LAN techniques. A plurality of applications can also be run by system 10 using well-known techniques.

When application 15 sends a request or system call to kernel 13 for opening, reading, writing, closing, etc., a file, for example, the request is received by a logical layer 22 that includes shared library 16 and access server 17. Preferably, shared library 15 presents applications with a complete POSIX system interface. Shared library 16 is linked by application 15 and runs in the address space of application 15. All of the necessary functionality for handling system calls is provided by shared library 16. According to the invention, shared library 16 includes a capability for handling a PNS, that is, shared library 16 is capable of determining whether the pathname for a file specified in

a system call is located under a PNS for a user of the application. The determination is made by examining the mount point (the iroot). If the specified file is located under a PNS, shared library 16 determines whether the requested file is an Internet file or is stored locally. If the file is an Internet file, shared library 16 communicates the request from the application 15 to access server 17 through an interprocess communication (IPC) in a well-known manner. For computing systems that provide dynamic shared libraries, application 15 may invoke shared library 16 by simply redefining a library search path to include shared library 16. For computing systems without dynamic shared libraries, applications must be relinked with shared library 16.

Access server 17 selects the appropriate access protocol for the specified file, connects to remote server 19 via the Internet 18, and obtains the specified file which is stored in memory 20 from remote server 19. Access server 17 is capable of performing an authenticated access of an Internet server "on behalf of" a user who is running application 15. That is, access server 17 can access remote file servers using the user's password information. To do this, shared library 16 is capable of authenticating itself to access server 17.

Once the specified file is obtained, access server 17 caches the file locally in cache 14. Access server 17 contains a plurality of agents, with each agent implementing a particular access protocol. Figure 1 shows exemplary agents 17a, 17b and 17c. Agents may contact remote servers directly or through proxy servers, such as proxy server 21. Access server 17 performs three basic functions of mounting a remote file system on a local

name space, retrieving a remote file and restoring a file to its primary server. The functions of retrieving and restoring are mapped to corresponding commands supported by specific protocols, while the function of mounting is a local operation.

Access server 17 maintains a mounting database for each PNS defined within system 10. Information required for accessing an Internet file, such as the access protocol, the host, authentication information, remote path information, etc., are stored in the database. Each entry for a PNS also includes the mount point for the PNS. Access server 17 refers to the mounting information in the mounting database for locating, retrieving and storing the file in cache 14 of the operating system kernel 13. Access server 17 sends an IPC message to shared library 16 when the specified file has been cached. Shared library 16 then retrieves the specified file from cache 14. When the filed is closed by application 15, access server 17 restores the file back to remote server 19. When the specified file is a local file, that is, stored locally elsewhere within system 10, shared library 16 directly accesses the file.

Preferably, an Internet file is cached in its entirety on the local file system when it is opened because remote file servers are contacted only for system file "opens" and "closes" and not for individual reads and writes. In this manner, the total network overhead incurred by transmitting a file is lower when the entire file is retrieved rather than as a series of system requests and responses for individual pages. Since accesses to files over the Internet are expensive in terms of overall system resource consumption, when a cached

copy of a file is accessed, the present invention determines whether the data of the cached are consistent with master copy data stored in the remote server.

While a single access server 17 is shown in layer 22 in Figure 1, a plurality of access servers can be used. For example, a basic configuration of the present invention can include an access server 17 for each system user and a system wide name server for locating each user's access server. Alternatively, groups of users can share a particular access server 17. The present invention also allows file servers located on the Internet to be mounted as user-level implementations, as shown in Figure 2, rather than as a kernel implementation as in conventional systems. In Figure 2, each shared library 16 that is associated with a process 23 directly accesses access server 17, rather than the application being processed through an associated kernel, the NFS protocol (not shown) and an access server (not shown), as is presently done in the prior art.

Figure 3 illustrates the concept of a PNS within a local file system 30. A first PNS 31 is shown mounted to local file system 30 at /home/bob/IFS and owned by the user Bob. A second PNS 32 is shown mounted at /home/stephanie/tmp/IFS and owned by the user Stephanie. A PNS is defined by the directory pair (MountPoint, CacheDirectory), where MountPoint is a local pathname in the file system to which PNS is mounted or attached, and CacheDirectory is a local directory in which remote files and directories are cached. Essentially, MountPoint indicates where the mount is pointing. In Figure 3, the mount point for PNS 31 is /home/bob/IFS and the

mount point for PNS 32 is /home/stephanie/IFS. Cachedirectory is needed in a file system structure as a place to put a cache file. To simplify the task of maintaining multiple PNSs, the present invention uses a single directory called an IRoot for a MountPoint and a CacheDirectory. An IRoot is a mount point for a PNS and a physical directory on the local file system where remote files and directories are cached. That is, an IRoot includes the mount point and the cache directory. A user defines a PNS by specifying the IRoot using the command iroot:

$ iroot pathname

As shown in Figure 3, the IRoot for PNS 31 is defined to be /home/bob/IFS, while the IRoot for PNS 32 is defined to be /home/stephanie/tmp/IFS. PNSs 31 and 32 are each mounted on local name space 30. Files and directories located under PNS 31 (/home/bob/IFS) are accessible only to the particular user defining PNS 31. To other users, such as the user who defined PNS 12, the files and directories under /home/bob/IFS are undefined.

The Universal Resource Locator (URL) naming scheme is embedded within the system of the present invention. The URL naming scheme has the form:

protocol://user:password@host:port/path

where "protocol" is the access protocol, "user" is an optional user name, "password" is an optional password, "host" is the fully qualified domain name of a network host or its IP address, "port" is an optional port number to which

to connect to, and "path" is the path used by the remote server for accessing the desired resource. The access protocol not only provides the key to accessing Internet resources located on remote servers, but also hides the heterogeneity of a system on which the accessed server is located. That is, once an access protocol becomes available, it is possible to access resources provided by a server using the protocol without regard to the machine type or the operating system of the server. Consequently, the present invention supports a heterogeneous collection of access protocols.

Table I shows exemplary pathnames and corresponding URL names with "–IFS" designating the IRoot that includes the mount point and cache directory.

TABLE I

IFS Pathname	URL Name
~/IFS/ftp:/ftp.ai.mit.edu/pub/README	FTP:/ftp.ai.mit.edu/pub/README
~/IFS/rsh:/king/src	rsh://king/src
~/IFS/http:/research.att.com/people/list.html	http://research.att.com/people/list.html
~/IFS/nntp:/ulysses/comp.os.research/5364	nntp://ulysses/comp.os.research/5364
~/IFS/gopher:/csie.nctu.edu.tw	gopher://csie.nctu.edu.tw
~IFS/ftp:/john:xxx@gryphon/home/john	ftp://john:xxx@gryphon/home/john
~/IFS/http:/jade:8000/project/DAF.html	http://jade:8000/project/DAF.html
~/IFS/http/192.127.159.32:8000/IFS.html	http://192.127.159.32:8000/IFS.html

The following exemplary pseudo code opens a home page using the system call open(), reads a file located at a URL name using the system call read(), processes the contents of the file as raw data, and closes the file using the system call close():

```
fd = open("~/IFS/http:/research.att.com/books.html",
0_RDONLY);
while ((num = read(fd, buf, nbyte)) > 0)
{
        /* process data in buf */
}
close(fd)
```

The present invention also provides that an Internet resource can be located, managed and interacted with using existing commands and tools. For example, the command line

$ ls ~/IFS/nntp:/ulysses/comp.os.research/*

lists articles in the news group "comp.os.research" that are stored on the News server "ulysses." Additionally, the command line

$ grep thread ~/IFS/nntp:/ulysses/comp.os.research/*

searches for the keyword "thread" in the News group "comp.os.research".

Mounting allows shortcuts to be created when naming files by providing indirect naming and also maintains authentication information for contacting remote servers. Additionally, mounting allows a proxy server to be specified through which the present invention accesses a desired file server. Both PNSs and mount points within a PNS are defined on a per-user basis. According to the invention, a mount operation allows a user to attach a name space, such as PNS, to another name space. The mount operation can attache a PNS on top of a global name space, and can attach a remote file system to a PNS.

A mount point is specified by the command mount:

$ mount URLName MountPoint [ProxyServer:Port] [ValidPeriod]

where "URLName" is the URL name of a file/directory accessible over the Internet, "MountPoint" is the defined mountpoint in a PNS (for example, IFS in

Figure 3), "ProxyServer" is an optional proxy server parameter, and "Port" is an optional parameter specifying the tcp port of the proxy server.

When the ProxyServer and Port parameters are specified, the present invention accesses the specified remote server through the specified proxy server. The parameter ValidPeriod (in seconds) specifies a time period during which cached files under the mount point are considered to be valid. Specifically, when a file is accessed, the cached file can be used when its age is still within the specified ValidPeriod. If ValidPeriod time has expired, the file must be obtained from the network. The URLName may contain a user name and a password, and when present, the present invention authenticates access to the remote server on behalf of the user using the password. If the value of password is "-", the mount command preferably requires the user to enter the password at the next prompt.

Figure 4 illustrates how remote file systems are mounted according to the present invention. In Figure 4, remote file systems that use different protocols are mounted to the IRoot /home/bob/IFS (PNS 31 in Figure 3). File server 40 at gryphon.att.com is mounted to /home/bob/IFS as follows:

$ mount ftp://bob:-@gryphon.att.com.home/bob /home/bob/IFS/research

Password: ******

This command mounts the directory bob, located in the file server 40 on gryphon.att.com, on PNS 31 at the node ~/home/bob/IFS/research using the ftp protocol. In this example, the command mount prompts Bob to enter a

password because "-" has been specified in the password field. Whenever the user Bob access files under /home/bob/IFS/research, the present invention retrieves the corresponding files from gryphon.att.com on behalf of Bob, using Bob's entered log-in and password. Another example of the mount command is:

$ mount rsh://king/home/rao/project /home/bob/IFS/project

For this example, the directory project, located in the file server 41 at king, on PNS 31 at the node ~/home/bob/IFS/project via the rsh protocol, which provides authenticated accesses. Users specify user names and passwords inside URLName. The system takes authentication information from the .rhosts file in the user's home directory. This avoids passing password information through the network.

In the following example of the mounting command, a mount point is created using an anonymous ftp access:

$ mount ftp://tsx-11.mit.edu/pub/linux /home/bob/IFS/src/linux radish:8000

where the proxy server on radish on the port 8000 is used for accessing tsx-11.mit.edu. The remote directory linux in the file server 42 on tsx-11.mit.edu for this example is mounted on the node linux in PNS 31.

The present invention provides two methods for controlling cache validation processes. First, an attribute ValidPeriod, assigned using mount commands, is associated with each directory. Each directory inherits VaildPeriod from its parent directory. A cached copy of a file under a created

directory is treated as valid, without checking its primary copies, if the copy was retrieved during ValidPeriod. For example, if ValidPeriod for a cached file is set to be 7200 seconds, the cached copy is considered as valid for 7200 seconds. When ValidPeriod is "-1", the cached copy is always considered valid. When ValidPeriod is "0", the cached copy is always invalid. ValidPeriod can be changed dynamically like many other attributes associated with directories.

For the second validation control method, a per-process mask, referred to as ValidMask, is provided for overwriting the ValidPeriod of a directory. A child process inherits ValidMask from its parent process and a built-in shell command vmask is used for defining and updating the process ValidMask. The per-process ValidMask has higher priorities than the per-directory ValidPeriod. That is, when dealing with a particular type of file, the valid time is determined according to the mounting, i.e., the file desired to be accessed. When considering the purpose for using the file, the valid time is the time of the process according to the application, i.e., the time the application uses the file. Thus, with ValidPeriod, cache validation processes can be controlled according to types of Internet files, such as co-authored papers versus Network News, whereas, ValidMask allows overwriting ValidPeriod on the process level in special circumstances, such as refreshing cache 14.

Figure 5 shows a flow diagram for a validation process 50 according to the present invention. At step 51, it is determined whether the requested information is in the cache. If the requested information is in the

cache, flow continues to step 52 where it is determined whether the cache is valid under ValidPeriod or ValidMask. If so, flow continues to step 53 where the information in the cache is retrieved for the system call. If the information is not in the cache (step 51) or the cache is not valid (step 52), flow continues to step 54 where a message is sent from the shared library to the access server. Flow continues to step 55, where the Internet is accessed, the information is retrieved and then is stored in the cache.

Figure 6 shows a flow diagram 60 of the details of step 52 of process 50 shown in Figure 5. At step 52A, it is determined when the cache was last updated. Then, at step 52B, the last update time is compared with ValidPeriod or ValidMask for determining whether the cached copy is valid.

The present invention identifies itself to a server for authenticating the user. Consequently, shared library 16 and access server 17 are preferably configured inside system 10 as a separate entity. In this way, shared library 16 is aware of the identity of a user. Access server 17 obtains information from shared library 16 for informing servers connected to the Internet because access server 17 is not configured for determining the identity of the user making a system call request. Consequently, shared library 16 identifies itself to access server 17 before the access server can perform user authentication.

The present invention also supports both anonymous and authenticated accesses to Internet servers directly or via proxy servers. According to the invention, users can authenticate themselves to remote servers

using a proper login name and password information for accessing files. An authenticated access includes a handshaking process in which shared library 16 first communicates with access server 17 that a request is to be authenticated. Access server 17 then communicates to shared library 16 that a file is to be created for obtaining a time stamp. Shared library 16 identifies the file to access server 17 once the file has been created. Access server 17 then accesses the file created by shared library 16 for confirming the information.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.